Docket No. 042390.P11634

United States Patent Application

in the Name of

Nagasubramanian Gurumoorthy

and

Raul Yanez

for

FAULT RESISTANT OPERATING SYSTEM

Submitted by

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP

12400 Wilshire Blvd., Seventh Floor

Los Angeles, CA 90025-1026

BACKGROUND

Field:

[001] The subject matter disclosed herein relates to computing systems. In particular, the subject matter disclosed herein relates to computing systems which comprise operating systems.

[002] Information:

[003] A processing platform typically comprises a central processing unit (CPU) which executes an operating system to control the use of resources for executing processing tasks. A processing platform typically initiates the execution of an operating system from machine-readable instructions maintained in a storage device in response to a reset event or a boot process. In response to a reset or boot event, the machine-readable instructions may be loaded to a system memory from a non-volatile storage device and then executed by the CPU from the system memory to "launch" the operating system.

[004] For any particular processing platform, more than one operating system vendor may develop of an operating system for the processing platform. An operating system may be developed to be hosted on any one of multiple processing platforms and a processing platform may be developed to host any one of multiple operating systems. Additionally, any particular operating system vendor may provide multiple versions of the same operating system. Depending on a particular processing and particular operating system, such an operating system may fail to launch on the processing platform in response to a reset event or boot process.

BRIEF DESCRIPTION OF THE FIGURES

[005] Non-limiting and non-exhaustive embodiments of the present invention will be described with reference to the following figures, wherein like reference numerals refer to like parts throughout the various figures unless otherwise specified.

[006] Figure 1 is a schematic diagram illustrating a computer architecture according to an embodiment of the present invention.

[007] Figure 2 is a schematic diagram illustrating a software configuration comprising a firmware interface according to an embodiment of the present invention.

[008] Figure 3 is flow diagram illustrating a process of loading an operating system to a processing system according to an embodiment of the present invention.

[009] Figure 4 is a flow diagram illustrating processes of loading an operating system to a processing system according to an embodiment of the process illustrated in Figure 3.

[0010] DETAILED DESCRIPTION

[0011] Reference throughout this specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the appearances of the phrase "in one embodiment" or "an embodiment" in various places throughout this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be combined in one or more embodiments.

[0012] "Machine-readable" instructions as referred to herein relates to expressions which may be understood by one or more machines for performing one or more logical operations. For example, machine-readable instructions may comprise instructions which are interpretable by a processor compiler for executing one or more operations one or more data objects. However, this is merely an example of machine-readable instructions and embodiments of the present invention are not limited in this respect.

[0013] "Machine-readable medium" as referred to herein relates to media capable of maintaining expressions which are perceivable by one or more machines. For example, a machine-readable medium may comprise one or more storage devices for storing machine-readable instructions. However, this is merely an example of a

machine-readable medium and embodiments of the present invention are not limited in this respect.

[0014] "Logic" as referred to herein relates to structure for performing one or more logical operations. For example, logic may comprise circuitry which provides one or more output signals based upon one or more input signals. Such circuitry may comprise a finite state machine which receives a digital input and provides a digital output, or circuitry which provides one or more analog output signals in response to one or more analog input signals. Also, logic may comprise processing circuitry in combination with machine-executable instructions stored in a memory. However, these are merely examples of structures which may provide logic and embodiments of the present invention are not limited in this respect.

[0015] A "processing system" as discussed herein relates to a combination of hardware and software resources for accomplishing computational tasks. However, this is merely an example of a processing system and embodiments of the present invention are not limited in this respect. A "host processing system" relates to a processing system which may be adapted to communicate with a "peripheral device." For example, a peripheral device may provide inputs to or receive outputs from an application process hosted on the host processing system. However, these are merely examples of a host processing system and peripheral device and embodiments of the present invention are not limited in these respects.

[0016] A "data bus" as referred to herein relates to circuitry for transmitting data between devices. For example, a data bus may transmit data between a host processing system and a peripheral device. However, this is merely an example of a data bus and embodiments of the present invention are not limited in this respect. A "bus transaction" as referred to herein relates to an interaction between devices coupled in a bus structure wherein one device transmits data addressed to the other device through the bus structure.

[0017] An "operating system" as referred to herein relates to one or more encoded procedures for facilitating communication between application procedures and processing resources of a processing system. Such an operating system may allocate processing resources to application procedures and provide an application programming interface (API) comprising callable software procedures for execution on the processing resources in support of application procedures.

However, these are merely examples of an operating system and embodiments of the present invention are not limited in these respects. A "basic input/output system" (BIOS) refers to systems for providing machine-readable instructions ("BIOS routines") to a processing system processor for initializing hardware resources of a processing system.

[0018] A "system reset" as discussed herein relates to an event or procedure in which a processing system is returned to a predefined state. For example, a system reset procedure may involve halting execution of one or more processes on a processing system followed by executing a process to attempt loading an operating system. This may be preceded by an execution of instructions provided by a BIOS to retrieve instructions for the operating system from a non-volatile memory. However, this is merely an example of a system reset procedure and embodiments of the present invention are not limited in this respect.

[0019] A "firmware interface" refers to software routines and data structures to enable communication between an operating system and hardware resources of a processing system. Such a firmware interface may define an interface between the hardware resources of a processing system and for one or more or more independently developed operating systems. However, this is merely an example of a firmware interface and embodiments of the present invention are not limited in this respect. According to an embodiment, BIOS routines may be executed on hardware resources to install the software routines and data structures of a firmware interface on hardware resources of a processing system and then subsequently install an operating system during a boot sequence or in response to a system reset event. However, this is merely an example of a firmware interface and embodiments of the present invention are not limited in this respect.

[0020] A "boot procedure" as discussed herein relates to a procedure by which initial instructions are loaded to a memory of a processing system and then executed. For example, a boot procedure may involve loading an operating system to a memory of a processing system. This may be preceded by the execution of instructions provided by a BIOS associated with the processing system which instructs the processing system to retrieve instructions from a non-volatile memory device for providing an operating system. However, this is merely an example of a boot procedure and embodiments of the present invention are not limited in this respect.

[0021] A program "launch" as referred to herein relates to an initiation of an execution of the program on a processing system. This may occur, for example, upon an initiation of the execution of instructions for the program at a location in a system memory of the processing system. However, this is merely an example of a program launch and embodiments of the present invention are not limited in this respect.

[0022] A "boot manager" as referred to herein relates to logic or data in a firmware interface defining a sequence of events or actions to be taken in response to a system reset event on a processing system. A boot manager may define variables that point to files to be loaded in response to a system reset event. For example, a boot manager may enumerate one or more processes for loading an operating system to the processing system. Additionally, a boot manager may initialize essential drivers and determine available operating systems for the processing system. However, these are merely examples of a boot manager and embodiments of the present invention are not limited in these respects.

[0023] Briefly, an embodiment of the present invention is directed to a system and method of launching an operating system on a processing system. A firmware interface may be launched on a processing system. The firmware interface may comprise logic to attempt launching an operating system on the processing system. Upon detection that the attempt is unsuccessful, a system reset on the processing system may be automatically initiated. However, this is merely an example embodiment and other embodiments of the present invention are not limited in these respects.

[0024] Figure 1 is a schematic diagram illustrating a processing platform architecture 20 according to an embodiment of the present invention. A central processing unit (CPU) 2 is coupled through a data bus 10 to a random access memory (RAM) 4, basic input/output system (BIOS) 6 and a non-volatile memory (NVM) 8 such as a hard disk drive or flash memory device. Devices on the bus 10 may also be coupled to one or more peripheral devices such as a network interface controller (NIC) 18, universal serial bus (USB) 16 and small computer system interface (SCSI) 22 through a bridge 14 and a bus 12. However, this is merely an example architecture of a processing platform and embodiments of the present invention are not limited in this respect. Also, the busses 10 and 12 may be any suitable

communication bus such as a peripheral components interconnection (PCI) bus. However, this is merely an example of a data bus format and embodiments of the present invention are not limited in this respect.

[0025] A "system memory" of the presently illustrated embodiment may comprise portions of the RAM 4 and NVM 8 to provide memory resources for use during dynamic operation of the processing system. In the illustrated embodiment, the BIOS 6 may comprise a memory for storing BIOS routines to be executed on the CPU 2 during a boot sequence or in response to a system reset event. Execution of the BIOS routines may initiate the loading of a firmware interface from the BIOS 6 or NVM 8 to the RAM 4, followed by the loading of an operating system from the NVM 8 to the RAM to create an image in the system memory. Following the boot procedure, the firmware interface may provide pointers to locations in the system memory to direct the CPU 2 to execute instructions in the image for performing tasks. However, embodiments of the present invention are not limited in this respect and a firmware interface and operating system may be loaded to a system memory using other techniques.

[0026] Such an operating system loaded to the RAM 4 during a boot procedure may comprise, for example, any one of several operating systems for desktop or mobile computers such as, for example, versions of Windows™ sold by Microsoft Corporation, or any one of several operating systems for real-time or embedded applications such as, for example, versions of Linux or versions of VxWorks or pSOS sold by Windriver Systems, Inc. However, these are merely examples of operating systems for desktop or mobile computer systems and for real-time or embedded applications, and embodiments of the present invention are limited in this respect as other operating systems may be used.

[0027] Figure 2 is a schematic diagram illustrating a software configuration comprising a firmware interface according to an embodiment of the present invention. In the illustrated embodiment, a system reset event or boot procedure may install a firmware interface such as an extensible firmware interface (EFI) as described in the Extensible Firmware Interface Specification, Version 0.99, April 19, 2000 published by Intel Corporation (hereinafter "EFI Specification"). However, embodiments of the present invention are not limited in this respect and other firmware interfaces may be used.

[0028] In the illustrated embodiment, an operating system may communicate with platform hardware 108 through an EFI or interfaces 112 for other services such as, for example, an Advanced Configuration and Power Interface (ACPI), ACPI Specification, Revision 1.0, December 22, 1996, Intel Corp., Microsoft Corp. and Toshiba Corp., and System Management BIOS (SMBIOS), SMBIOS Reference Specification Version 2.3.1, March 16, 1999. However, embodiments of the present invention are not limited in this respect and the operating system may communicate with system hardware using other techniques.

[0029] The platform hardware 108 comprises a system memory 118 which is capable of storing executable images of the operating system 102 and the EFI. In the illustrated embodiment, the EFI runtime services 106, EFI boot services 110 and EFI operating system (OS) loader 104 may reside in a first area 116 of the system memory 118 and the operating system 102 may reside in a second area 114 of the system memory 118. While Figure 2 shows that first and second areas 116 and 114 of the system memory 118 are contiguous, it should be understood by those of ordinary skill in the art that such areas of memory need not be physically contiguous in the system memory 118. It should be understood that locations internal to the area 116 need not be contiguous in the system memory 118.

[0030] According to an embodiment, BIOS routines may load the EFI runtime services 106 and EFI boot services 110 to the system memory 118 separately from a process for loading of the operating system 102. The EFI runtime services 106 and EFI boot services 110 may be launched to the platform hardware 108 in response to a system reset. Upon launching the EFI boot services 110, a boot manager may launch an EFI OS loader 104.

[0031] In the illustrated embodiment, an "EFI System Table" may be maintained in conjunction with the firmware interface to provide a reference to EFI boot services 110. The EFI System table may maintain a list of globally unique identifiers (GUIDs) referenced to function pointers. Accordingly, the EFI OS loader 104 may retrieve function pointers to the EFI boot services 110 using the GUIDs. However, embodiments of the present invention are not limited in this respect and pointers to runtime functions may be located using other techniques.

[0032] Figure 3 shows a flow diagram illustrating a process 200 for attempting to load an operating system to a processing system according to an embodiment of the

present invention illustrated with reference to Figure 2. At block 202, a system reset is initiated at the processing system. The system reset may be initiated by a BIOS routine in response to a boot procedure or processing system event. However this is merely an example of how a system reset may be initiated in a processing system and embodiments of the present invention are not limited in this respect. In response to the reset event, hardware in the processing system may be initialized at block 204 by, for example, one or more BIOS routines. Then, initialization of peripheral devices and allocation of resources to support the peripheral devices (e.g., bus enumeration) may be performed by BIOS routines at block 206. However, this is merely an example of how a processing system may be initialized in response to a reset event and embodiments of the present invention are not limited in this respect.

[0033] In addition to initializing peripheral devices and allocating resources to same at block 206, BIOS routines may launch a firmware interface such as an EFI as described in the EFI specification. For example, block 206 may comprise launching a boot manager as part of the EFI. In an embodiment of a processing system as illustrated with reference to Figure 2, such an EFI may comprise EFI boot services. The firmware interface may comprise a boot manager which facilitates launch of an OS loader at block 208. In the illustrated embodiment, the OS loader may comprise logic to launch one or more operating systems to the processing system. However, this is merely an example of how an OS loader may be initialized and embodiments of the present invention are not limited in this respect.

[0034] In the illustrated embodiment, a boot manager may launch any one of multiple OS loaders where each OS loader comprises logic to launch an associated one of multiple operating systems. The boot manager may determine which of the operating systems are capable of being hosted on the processing system base on, for example, the hardware configuration of the processing system as indicated by BIOS routines. The OS loader of a capable operating system may then be selected to launch the operating system. However, this is merely an example of how a boot manager may support the launching of an operating system selected from multiple operating systems and embodiments of the present invention are not limited in this respect.

[0035] At block 210, the OS loader may set a watchdog timer to a prespecified time interval, attempt to launch an operating system, and wait at diamond 212 for either a detection of a successful launch of the operating system at block 214 or an unsuccessful attempt at block 218. According to an embodiment, each OS loader for each operating system may specify a different time interval for the watchdog based on one or more factors such as, for example, particular characteristics of the operating and an expected time to launch, platform hardware characteristics and any particular operating requirements associated with the processing platform (e.g., embedded system requirements or desktop system requirements). Logic for determining the time interval may be provided in a BIOS routine. However, these are merely examples of how a time interval for a watchdog timer may be determined and embodiments of the present invention are not limited in these respects.

[0036] Block 218 detects an unsuccessful attempt to launch when the watchdog timer expires before the operating system has been launched (i.e., the processing system is considered to be "frozen"). Upon detection of such an unsuccessful attempt, block 220 initiates a system reset at block 202. Otherwise, if the operating successfully launches before the watchdog timer expires, block 214 may disable the watchdog timer and terminate the OS loader before the operating system takes control of the processing platform at block 216. In the illustrated embodiment, block 214 may detect a successful launch of an operating system by, for example, detecting the completion of one more tasks initiated by the OS loader and the absence of one or more error conditions. However, this is merely an example of detecting a successful launch of an operating system and embodiments of the present invention are not limited in this respect.

[0037] According to an embodiment, the boot manager may comprise logic to attempt loading different operating systems on subsequent reset cycles. Therefore, if an attempt to launch a first operating system on a first reset cycle may be followed an attempt to launch a second, different operating system on a second reset cycle if the first attempt is unsuccessful. Accordingly, the OS loader at block 210 may set the processing system to attempt launching a second operating system in a subsequent reset cycle if an attempt to launch a first operating system in the current reset cycle fails. Upon successful launch of the operating system, the OS loader at block 214

may then set the boot manager to attempt launching the first operating system in a subsequent system reset cycle or system boot. However, this is merely an example of how a firmware interface may enable attempts to launch an alternative operating system if an attempt to launch an initial operating system fails.

[0038] Figure 4 shows a flow diagram illustrating a process 300 according to an embodiment of the process illustrated in Figure 3 in which the boot manager and OS loader are provided in a firmware interface according to the EFI specification. At block 302, the OS loader may set a watchdog timer as illustrated in the EFI specification at Section 3.8.2. In one embodiment, a particular OS loader for an operating system may determine the prespecified time delay for waiting for a successful OS launch depending on the particular operating system for the current launch attempt, the processing platform configuration or a particular purpose of the processing platform. However, this is merely an example of setting a watchdog timer to detect an unsuccessful attempt to launch an operating system and embodiments of the present invention are not limited in this respect.

[0039] The boot manager may define "BootOrder" as an array of operating system identifiers associated with operating systems (i.e., operating systems for which launches may be attempted), "BootCurrent" as an index to an identifier for the currently selected operating system identified in the BootOrder array (i.e., the operating system for which a launch may be attempted in the current system reset cycle) and "BootNext" as an index to identifier in the BootOrder array for the next operating system to be selected for a subsequent reset cycle if the operating system identified by BootCurrent fails to launch in the current reset cycle. The OS loader may then define which operating system is to be launched in a subsequent reset cycle by setting BootNext as the index in the BootOrder array to the currently selected operating system incremented by one.

[0040] At diamond 312, the OS loader may detect a successful launch of the operating system defined by, for example, completing one or more tasks initiated by the OS loader in the absence of one or more particular errors. Upon detection of a successful operating system launch, block 314 may disable the watchdog timer by through SetWatchDogTimer and delete BootNext to ensure that the successfully loaded operating system is loaded in response to a subsequent reset event.

[0041] While there has been illustrated and described what are presently considered to be example embodiments of the present invention, it will be understood by those skilled in the art that various other modifications may be made, and equivalents may be substituted, without departing from the true scope of the invention. Additionally, many modifications may be made to adapt a particular situation to the teachings of the present invention without departing from the central inventive concept described herein. Therefore, it is intended that the present invention not be limited to the particular embodiments disclosed, but that the invention include all embodiments falling within the scope of the appended claims.